The use of ALLIANCE in Robot-Human Interaction

Jeanette Booher and Katie Creel[1]

I. Introduction

Between the remote-controlled simple robots of the past and the fully independent robots of the movies stands our current problem: autonomous robots that cooperate with humans to perform specified tasks. Such robots are able to function and make decisions without explicit commands from the human. In many situations where robots and humans might be working together, wireless communication would be impractical or impossible, such as when the team is working outdoors or in certain dangerous situation, such as hazardous waste cleanup. In military applications, the wireless might be down or jammed. Voice communication could also prove problematic if the human was too far away, or in too noisy an environment. Lasers and cameras, comparatively, are much more robust.

ALLIANCE, a control architecture for multi-robot cooperation developed by Dr. Lynne Parker [1], fits the needs of a robot in such a situation ideally. ALLIANCE focuses on increasing fault tolerance and eliminating the need for communication. Furthermore, the manner in which the robot works is independent. Thus although more trust is placed in the human's ability to judge the robot's capabilities and success rate, it and the human are effectively peers who both independently choose subtasks to perform. We modified ALLIANCE and extended it to take into account the new parameters necessary for effective robot-human cooperation. Our experiments focus on cooperation between one robot, a Pioneer P3-DX, and one human, but the algorithm could easily be extended to multi-human and multi-robot teams. We chose to test this control architecture on the box pushing task, which is a standard benchmark, and used an overhead camera to identify boxes and help the robot choose which box to push.

II. Box Identification

In order for the robot to use ALLIANCE to assist the human in the box pushing task, we needed to find or create software that would work with an over head camera to identify boxes in the robots environment and then send that information to the robot.

To process images from the camera, we used OpenCV [2], which is an open source image processing software created by Intel. Fortunately, OpenCV had some example code which was able to find rectangular objects within an image. The code provided worked by taking an array of images. The program then looped through each image and its three color planes. The algorithm used Sobel line detection and a polygon approximation function to find all the possible polygons within the image. All of the possibilities were then checked to ensure that each had exactly four vertices, was larger than a small minimum area, and was convex in nature. The polygons that satisfied all of these constraints were then passed to a function in which they would be drawn and displayed on the screen.

To tailor the code to our goals, we had to heavily modify the example code. One of the first modifications was to have the function capture the images it would analyze from a web camera. Because the original algorithm had difficulty isolating real boxes from noise within the environment, we added a training period. During this time, images of a box less environment are analyzed using the same basic polygon detection as the original algorithm and each polygon that is identified is marked as noise. Training lasts for three seconds and is only conducted once. After training, the new

[1] This paper was a joint collaboration between Jeanette Booher and Katie Creel. Sections II, III, V, VI, References and Acknowledgement were completed by Booher while sections I and IV were completed by Creel.

images that are fed into the program are also passed though the original polygon detection algorithm. The possible boxes are then checked to ensure that they are not part of the environment and that they have not already been identified in previous frames of the current iteration. The points of each rectangle are then organized into a box with a top left, top right, bottom left and bottom right corner. This is done by comparing the distances to zero of the points and then comparing unknown points to identified ones. Area is used to identify truly rectangular polygons. The area created by multiplying the length of two adjacent sides of the box and the area of the opposite adjacent sides are compared. If the areas are similar and the box is smaller than a third of the frame size, the box is saved. At the end of a five second period, the pixel information for each corner of the box is converted into the global coordinate system of the robot. Because our camera was placed perpendicular to the floor, this was done with a simple ratio which converted pixel value to distance from the (0, 0) pixel and then the added the coordinate value within the robots global coordinate system to that value. The information is then sent to the robot using a socket, where it can be analyzed by ALLIANCE in order to accomplish a chosen box pushing task.

III. Box Code Testing

Overall the box finding code proved to be fairly accurate. Over eight trials with contained a total of 40 boxes, the algorithm identified 54 possible boxes. There were no false negatives, but there were nine false positives and five non unique boxes identified. This data is displayed in Figure1.1. The false positives were mainly due to shadows

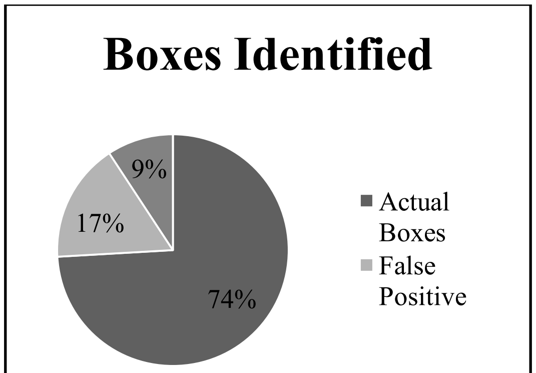

Figure 1.1

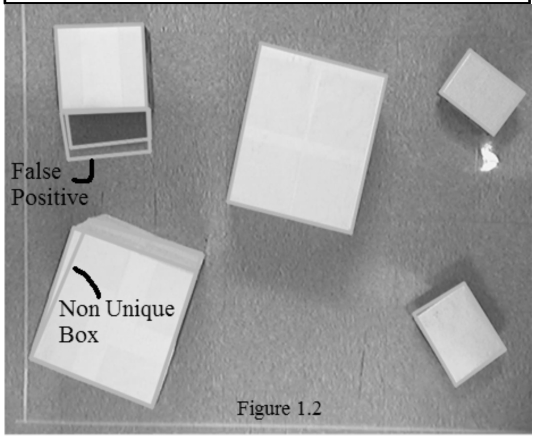

Figure 1.2

created by the boxes on the floor, and the non unique boxes were identified because of how our algorithm checked for box uniqueness. Figure 1.2 illustrates both of these problems. These problems arose because the algorithm being used for testing only compared one point of the possible boxes against all four points of the established boxes to check for box uniqueness. We have since changed the algorithm to compare all four points of a possible box, but were unable to collect additional box identifying data before the end of the DREU. Personal observations did however show that the number of false positives seemed to have decreased substantially.

IV. ALLIANCE in HRI

Our decision architecture is a modified form of ALLIANCE, a fault tolerant architecture for multi-robot cooperation. ALLIANCE's main component is the motivational behavior set, which contains a behavior set. The behavior set describes a possible subtask, which the robot could select, and the motivational behavior set, which wraps around it, contains the information necessary to calculate the robot's motivation to perform that task. By comparing its motivation for each behavior set, the robot is able to choose the best task to perform.

The core of ALLIANCE is the calculation of motivation, which we altered from the original calculation, which was based on multi-robot interactions. The original motivation calculation is shown in Figure 1.3.a. The impatience to perform a task grows slowly if another robot is attempting the task and quickly if it is undone. Acquiescence implies that the robot will give up more quickly if another robot is waiting, and after a certain time will acquiesce in acknowledgement that the task is taking too long and errors may have been encountered. Activity suppression becomes zero for a short period of time if a different task is chosen, giving another robot a chance to perform the task. Impatience reset ensures that motivation is reset when another robot attempts the task for the first time but not on a robot's subsequent attempts, so that a failing robot is not allowed to continually re-try the same task. And finally, sensory feedback becomes zero if the behavior set is not applicable to the current mission.

Although clearly applicable to the problems of human-robot interaction, ALLIANCE required adaptation to better fit the altered requirements. First, the human is presumed to be better able to judge the feasibility of the task she is working on. The robot therefore will not be impatient to take over the task if it takes the human longer than expected. Second, and more fundamentally, the robot and the human will not wirelessly communicate and the robot will have no prior knowledge of which task it is to perform. Therefore, the robot must analyze the human's behavior to ascertain what task the human is attempting before using ALLIANCE to determine which of the possible helpful sub-tasks it should perform.

$$M_t = [M_{t-1} + i] \times q \times s \times r \times sf$$

Figure 1.3.a

$$M_t = [M_{t-1} + i] \times q \times s \times c$$

Figure 1.3.b

The values of the given variables are defined as follows: $\boldsymbol{M}$ refers to motivation at the given time, $\boldsymbol{i}$ representing the impatience, $\boldsymbol{q}$ equaling acquiescence, $\boldsymbol{s}$ evaluating to activity suppression, $\boldsymbol{r}$ used for impatience reset, $\boldsymbol{f}$ for sensory feedback, and $\boldsymbol{c}$ defined as completion.

Thus the new algorithm is given in figure 1.3.b. Impatience here is a constant factor increase, from which is subtracted the relative distance away from the location of the task or box (absolute distance/mean distance). The acquiescence is based solely on time spent on task, in a manner similar to that of the previous slow acquiescence. Activity suppression is the same as in the robot teams. And completed merely ensures that robot will not re-attempt a task.

V. ALLIANCE Experiments

The ALLIANCE algorithm was first tested in simulation using Player/Stage [3]. In simulation, the robot was expected to move from its current position to the box that was closest to its current position. The robot then tagged the box as being complete and moved to the next closest box. After running multiple simulations with various numbers of boxes and with the boxes in diverse locations, the robot consistently moved from box to box in an order relative to the robots distance to the box. This is what we expected and reaffirmed the accuracy of the ALLIANCE algorithm.

The code was then tested on the Pioneer with calculated locations of white paper that had been placed on the floor which were sent directly to the Pioneer. The robot performed much as it had in simulation with few mistakes and was able to navigate to each square while avoiding unplanned obstacles as well as walls.

The last step that we completed was having the robot go to the locations of paper that were sent to the Pioneer via a socket from the over head camera system. The robot performed adequately and was able to maneuver easily to each squares location. The Pioneer simply tagged each box because the addition of pushing was not added before the end of the DREU.

I. Conclusion

These tasks are part of a larger project that's goal is to observe the behavior of a human working towards one of a set of predefined box pushing tasks and then identify what task the human is trying to accomplish. A camera capturing location information about boxes will then transmit this information to the robot, which will then use ALLIANCE to act in assist the human in the task that they are trying to accomplish. The identification of the human's task is currently being worked on by other members of the lab; this piece along with the refinement of the ALLIANCE code and the addition of box information like color and orientation to the camera sensing will hopefully be completed before the end of the year.

Once the code is operational, the goal will be to test the robot over numerous trials with an actual human working toward one of a set of predefined box pushing goals. The robot will then assist the robot using ALLIANCE and also using random box selection. The humans experience with the robot helper will then be evaluated with a questionnaire. The results of these questionnaires will hopefully show that the robot was able to help the human more through ALLIANCE than by random selection. It is our hope that the research we have done this summer will be helpful to other future researchers who are interesting in human-robot interaction and cooperation.

ACKNOWLEDGEMENT

The authors would like to thank Prof. Lynne Parker for her guidance and mentoring of this project and paper. We would also like to thank Richard Edwards, Yuanyuan Li, John Hoare, Sudarshan Srinivasan, Nick Overfield, and Tony Zhangya who also assisted with the project and helped in the learning of Player/Stage.

REFRENCES

[1] L. E. Parker. ALLIANCE: An architecture for fault-tolerant multirobot cooperation. IEEE Transactions on Robotics and Automation, 14(2):220–240, 1998.
[2] Bradski, G.R. (2001). Open Source Computer Vision Library. Intel Corporation.
[3] Collett, T.H., MacDonald, B.A., Gerkey, B.P.: Player 2.0: Toward a Practical Robot Programming Framework. In: Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005). (2005)